PSis - Programação de Sistemas 2014/2015
Exame 1ª Época, 27 de Junho de 2015, 11h30, Duração: 2h

Software Design

1. [1 value] One rule of systems' design is to **not overload the functionality of a component**.

- Describe what is excessive functionality overloading
- Explaining two problems that arise by not following it.
- Explain how this rule can be applied and implemented in a complex system.

A component of function implements several fiuunctionalities not necessarly related and answering to different requirements
Error prone code and difficult to maintain
Use one of the provided architectures and patterns with clear separation of reposability between the components

Software Architectures

2. [1 value] N-Tier architectures are widely used in web applications.

- Describe this architecture pattern.
- Explain how this architecture pattern affects the following system characteristics:
  - Maintainability.
  - Flexibility.

The system is separated into functional components that interact in a very strict pattern. Each tier only interact with two neighbor tier using well defined interfaces. Each tier is independent of the others
Since the interactions and responsibilities are well known and simple it is easy to find the errors and correct them changing the responsible tier.
The same characteristics also allow the change of a tier (with a different implementation, with different performance requirements without affecting the reaming tiers

3. [0.5 value] What is the main difference between a layered and a N-Tiered architecture?

A layered system runs in a single machine while a N-Tier system runs on different machines.

Software Testing

4. [1 value] The mathematical expression to calculate the real roots of a quadratic function is

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

It is to be implemented as a C function: `float * roots(float a, float, b, float c);`

- Define and describe the possible return values (corresponding to the float *).
- Define and describe a set of unitary tests that will allow verifying the corresponding code. It is not necessary to write the code but you have to describe what is to be tested and how.

The return float * will point to NULL, if no root exist or to an array of two floats (if the function only has one root both values will be equal).
The test suit should include the following tests:

- verify if two roots are found;
- verify if one root is found;
- verify if the function works correctly when no root is found.
- Verify if the function works correctly when a euqals 0

5. [1 value] Integration testing is one of the available test classes.

- Describe the uses of Integration testing.
- Describe when are these tests performed.
- Describe what class of error these tests are able to find.

Integration testing allows the correct construction of the systems from pre-programmed modules and allows the verification that the modules work correctly when integrated with the remaining system
These tests are performed after the unitary tests, where it is possible to perform the integration of several individual modules.
With these tests it is possible to test error related to the overall system requirements (that are fulfilled by different modules), it is possible to test the interfaces between modules

Multiprogramming

6. [1 value] In modern operating systems threads and processes allow multiprogramming.

- Describe two main differences between threads and processes.
- Explain why in a shared mail server each client should be assigned a different process and not thread.

Threads take less time to start, threads used less resources, threads share resources (memory) between them.
Processes offer isolation, so assigning a process to each client it is easier to offer privacy.

Inter-process communication

7. [1 value] FIFOs and message queues are mechanisms that allow inter-process communication.

- Describe one similarity between these two mechanisms.
- Describe one difference between these two mechanisms.
- Taking into account **extensibility**, which is the better of the two communication mechanisms (FIFO or message queues)?

Both use names to be identified by different processes, allow synchronization (when channel is full or empty).
Message queues have the concept of message, while FIFOs implement streams (no separation between different messages), the API is different.
FIFOS use a standard API (read/write) allowing the easy replacement by sockets.

8. [1 value] When implementation communication between components of a system (in the same machine or on different computers), for the correct functioning, there are some requirements that should be met.

- Describe three requirements that the communication protocol should follow.
- Describe how protocol buffers handle and ease the implementation of such requirements.

Correction Efficiency Interoperability (language/OS). Ease to use.
Protocol buffers allow programmer to define the message in a independent language, and create all the required code to create, process and verify the messages.

Synchronization

9. [2.5 values] Consider a company that manufactures items of type C from two other items of type A and B. In this company there are three warehouses, identified by ZA, ZB and ZC, which are used to store items of type A, B and C, respectively. The maximum capacity of the warehouses ZA, ZB and ZC, is NA, NB and NC, respectively.

There are several suppliers who put items of type A in ZA. Each supplier puts only one item, in the warehouse, at each time. The same applies to items of type B (which are put in ZB). The company works in continuous production cycles. In each cycle it produces an item of type C from a pair of items of type A and B, putting it in ZC. There are several vendors that take items of type C from ZC. Each vendor takes only one item C, from ZC, at each time.

Each warehouse should be accessed in mutual exclusion by the entities that use it. Considering the company, the suppliers and the vendors, distinct processes/threads, use semaphores (and only semaphores, not mutexes) with the common wait and signal operations to provide an implementation of the following functions:

- void putA() – executed by the suppliers of A
- void putB() – executed by the suppliers of B
- void putC() – executed by the company itself (where C is also manufactured by a pair of items A and B)
- void takeC() – executed by the vendors

Any of the following solutions were accepted.

Solution 1: each warehouse is not accessed with mutual exclusion.

```
sem_t cap_ZA=NA, cap_ZB=NB, cap_ZC=NC;
sem_t sem_A=0, sem_B=0, sem_C=0;

void putA(){
        wait(cap_ZA);
        //put item of type A
        signal(sem_A);
}

void putB(){
        wait(cap_ZB);
        //put item of type B
        signal(sem_B);
}

void putC(){
        wait(sem_A);
        signal(cap_ZA);
        wait(sem_B);
        signal(cap_ZB);
        //manufacture an item C from the previous pair of items of type A and B
        wait(cap_ZC);
        //put item of type C
        signal(sem_C);
}

void takeC(){
        wait(sem_C);
        //take item of type C
        signal(cap_ZC);
}
```

Solution 2: each warehouse is accessed with mutual exclusion.

```
sem_t cap_ZA=NA, cap_ZB=NB, cap_ZC=NC;
sem_t sem_A=0, sem_B=0, sem_C=0;
sem_t sem_access_A=1, sem_access_B=1, sem_access_C=1:

void putA(){
        wait(cap_ZA);
        wait(sem_access_A);
        //put item of type A
        signal(sem_access_A);
        signal(sem_A);
}

void putB(){
        wait(cap_ZB);
        wait(sem_access_B);
        //put item of type B
        signal(sem_access_B);
        signal(sem_B);
}

void putC(){
        wait(sem_A);
        wait(sem_access_A);
        //get item of type A
        signal(sem_access_A);
        signal(cap_ZA);

        wait(sem_B);
        wait(sem_access_B);
        //get item of type B
        signal(sem_access_B);
        signal(cap_ZB);

        //manufacture an item C from the previous pair of items of type A and B
        wait(cap_ZC);
        wait(sem_access_C);
        //put item of type C
        signal(sem_access_C);
        signal(sem_C);
}
```